# Pachyderm: The Web Proxy that Never Forgets.


Alison Krautkramer Jing Li Remzi Arpaci-Dusseau

sisko1@cs.wisc.edu jing@cs.wisc.edu remzi@cs.wisc.edu

Computer Sciences Department

University of Wisconsin

1210 West Dayton Street

Madison, WI 53705


December 18, 2000


**ABSTRACT**

As a result of the increasing popularity and resulting growth of the Internet; fast, efficient, and reliable access to the information on the Internet is becoming increasingly important. In this paper, an introduction to the Pachyderm Web Proxy is presented. This web proxy is an Internet proxy that not only caches the most recent copy of requested Internet documents, but it also stores old versions of those documents for future reference. Flexible context searches can be performed on the data within the cache to assist the user with identifying web pages of interest that have been viewed historically. This paper examines the design and implementation of the Pachyderm Web Proxy and provides measurements indicating that caching is an efficient method of satisfying client requests. Also, it will be demonstrated that indexing and searching historically viewed web documents can be done without the commitment of large amounts of memory or time.


## 1. Introduction

In recent years the Internet has exploded in popularity. In the past, the Internet was used almost exclusively for disseminating information in an educational or government setting, but now the Internet is becoming a standard in business. Finding that the Internet can lead to an even wider customer base, many businesses are investing significant resources in providing web pages that not only inform users but also sell goods and services.

The explosion of activity on the Internet has caused it to be a dynamic environment. A page that is viewed one day may be completely changed or deleted the next. In order to increase the reliability, speed and usability of the Internet we have devised a way to store historical copies of viewed web pages in a web proxy cache.

A web proxy is a server that acts as a middleman between a client and the Internet - figure 1.1. All requests from the client are sent to the proxy server instead of directly to the web server that serves the requested document. The web proxy uses the request to decide how to service the client in the fastest and most efficient way. First, the proxy will look for the requested document in a cache it uses to store copies of previously downloaded documents. If this document is found and it has not been changed since it was last downloaded; it is returned to back the user. However, if the document was not found in cache or it has been updated since the last time it was downloaded, then the proxy will forward the request to the appropriate web server for appropriate action. Upon receipt of the request, the external web server will fetch the correct document and send it back to the proxy. In turn, the document is summarily returned to the client. If the document can be cached, it will be stored in the proxy's cache in preparation for future client requests. Future requests for this document can then be served by returning the cached copy of the desired document instead of requiring contact from the external server and forcing the document to be downloaded again.

Most web proxy caches [1,4] only store the most recent copies of viewed web documents. As the user views new pages, the cache uses a page replacement policy to find an old web page in the cache to replace with the new information.

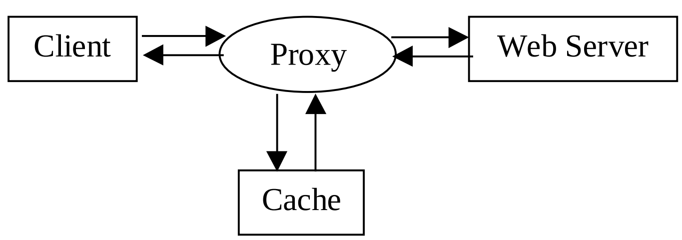


Figure 1.1 Web Proxy Server

In Pachyderm, all web pages that the user has viewed are stored in a cache. As a result, if a web page that the user is interested in is either removed from the Internet or the information it contained is changed, the user can still view the old copy of the page.

This paper will examine the design, implementation and evaluation of the Pachyderm Internet web proxy.  In Pachyderm, all copies of viewed web documents are stored in a cache. The cache can then be used not only to satisfy user requests in an efficient manner, but also serves to allow users the additional capability to view the contents of historical Internet documents even though those documents no longer exist.

## 2.  The Pachyderm Web Proxy

This section describes the implementation portion of the Pachyderm web proxy.

### 2.1 Communication: HTTP Headers

Communication among computers in the Internet is achieved through the use of HTTP messages. There are two types of messages in HTTP/1.1 and those are request messages and response messages - figure 2.1.


Figure 2.1: Downloading an Internet Document

Clients inform web servers of their request for a particular Internet document by issuing a request message.  A request message consists of a single line that contains information about the method to be applied to the source, the source requested, and the HTTP version being used followed by zero or more header fields that further describe the request.  There are six methods that can be specified in the request message: get, put, post, head, delete, and trace.  The Pachyderm web proxy only supports the get method and is described below.

The get method is a very important method from the perspective of an Internet web proxy.  Not only can a get method be used to request a desired document from the Internet, but the get method can be changed to a conditional get method with the addition of special header fields.  When a conditional get method is received by a web server that supports this method, the web server will only send the requested document back provided the document has been changed since the last time it was downloaded by the requestor.  Otherwise, if the document has not been updated, then only a response header stating that no modifications have been made is sent back. The purpose of the conditional get method is to improve the performance of Internet document caches. If the document has not changed, only a short message that indicates this fact will be sent back to the requestor instead of the entire document. As a result, a significant amount of time can be saved since a document can be retrieved from a local cache and returned to the client much faster than it could be read across a network.

### 2.2 Implementation

In order to implement the Pachyderm web proxy, first an open source Internet web proxy named RabbIT2 [6] was downloaded from the Internet. RabbIT2 served as the template on which Pachyderm is implemented.

Pachyderm was written and tested using Netscape Communicator as the client. Communicator was configured to send all of its requests for Internet documents to a pre-selected port; hence, Pachyderm was set up to listen for requests on this port and when requests were received it would create a thread to handle the request called a client connection - figure 2.2. This connection is used by Pachyderm to get a request from the client and then create a stream that can be used to send the document back to the client.

Pachyderm uses the request header sent by the client to perform a look up in a hash table of all the documents that are stored in the cache. If an entry is found in the hash table, then the name and path to the file in which the most recent copy of the document is stored, is returned. At the same time, a connection to the external web server of the desired document is created called a web connection. The web connection, like the client connection, stores the communication port between Pachyderm and the external server along with data streams that are used to send information to the external server and receive information from that server. Using the request header that was received the last time the document was downloaded from the Internet, a conditional get request is formed and sent to the server. If the server supports the conditional get method and the document has not been modified since the most recent copy in the cache, the server will respond with a header that indicates that the document has not been modified. Otherwise, if the web server does not support the conditional get method or the document has been modified, the web server will send back a response header followed by the contents of the document.

In a case where the web server sends back a response header indicating that the document has not been modified, the client connection gets the contents of the document from the cache file indicated by the hash table lookup. The contents will be sent along with a response header back to the client. However, if the contents of the document was returned by the web server, then the contents would be sent to the client and stored in a unique cache file within the cache directory that stores all of the former versions of this document.

If an entry does not exist for the requested document in the hash table, then there is no corresponding file in the cache. An entry will not be found in the hash table the first time a user requests any new document. In this case, the get request received from the client is forwarded to the external web server. The document contents that are received from the server will be placed in a file within a new directory created for this document. An entry will be placed in the hash table so that this document can be used to fulfill later client requests.

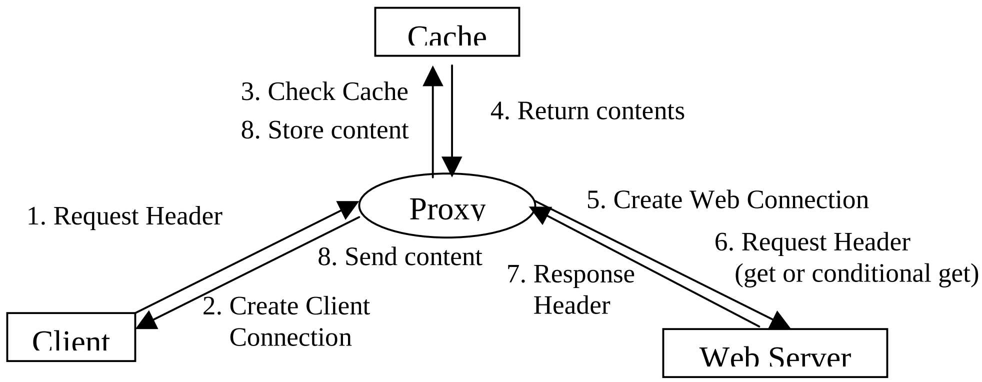


Figure 2.2: Pachyderm Structure

**2.3 Query Capabilities**

In order to search historically viewed Internet documents kept in Pachyderm's cache, a file system indexing tool named GLIMPSE [5] was used. GLIMPSE which stands for GLobal IMPlicit Search, is a tool that builds small indexes (usually 2-4% of the size of the original text [5]) that can provide support for boolean queries, approximate matching queries, and regular expressions.

In order to provide flexible support for querying text, GLIMPSE uses a two-level searching method that is a hybrid between a full inverted index and a sequential search. In order to build the index that GLIMPSE uses to answer queries, the files that contain text are broken up into blocks. The words that are found in each of these blocks are then placed in an index along with a pointer to the block of text where it came from. Each of the unique words found in a block are placed in the index once. Therefore, even if a word occurs several times within the same block of text, it will only be placed in the index once. This system helps keep the size of the index used by GLIMPSE small in magnitude.

Once the words within the file blocks are placed in the index, searches can be performed. GLIMPSE uses the search words that are provided by the user to sequentially search the index for matching words. When a matching word is found, the block of text containing that word can be accessed through the stored pointer. At this point, another sequential search is done on that block of text to exactly pinpoint where the desired word occurs. The phrase that the word occurs in can then be returned to the user along with the name of the file in which that phrase was found.

This tool along with an extension called WebGLIMPSE allows the user to be able to search an index built on the Internet documents stored in Pachyderm's cache via an HTML form on a web page. This application enables the user to have a simple and fast way to view Internet documents that they have viewed in the past.

## 3 Performance and Evaluation

This section describes the performance measurement that were performed the Pachyderm web proxy.

### 3.1 Caching Policies

In order to explore the advantages of caching along with the overhead incurred by storing all of the web documents that have been viewed by a user, two web proxies that use different caching polices were implemented. The first web proxy used a traditional keep one cache policy. In this version, only one copy of each requested Internet document would be stored in the cache. If that document were subsequently updated, then the new copy of the document would replace the old document. The second web proxy that was implemented is the Pachyderm web proxy that uses a keep all caching policy. In Pachyderm, not only is the current copy of an Internet document stored in the cache, but in addition, historically viewed Internet documents are also stored in the cache. Therefore, every cacheable web document that has ever been viewed by a client is stored within Pachyderm's cache.

### 3.2 Workload

One primary goal was to explore the effects of caching Internet documents on performance along with determining the overhead associated with the keep all caching policy. In order to achieve this goal, the authors attempted to minimize the fluctuations in the measurement caused by different-size documents and different loads on the web server; consequently, a workload was carefully designed to reduce noise that would affect the measurements.

To evaluate the performance of the Pachyderm web proxy and the traditional proxy, a workload consisting of a list of URLs (Uniform Resource Locators) was created. Each of these URLs was a plain-text document of a uniform size (~2.6 KB) that was stored on a local web server. We controlled this web server and were the only clients being served by it.

### 3.2 Average Response Time

To measure the average amount of time it takes the Pachyderm web proxy and the traditional web proxy to download an Internet document, a workload consisting of 10 uniform sized plain-text URLs was created and used. Each of these proxies was run with this workload and the average response time was measured. In this case, there is only one plain-text document for each URL, so the average response time is the average amount of time required to download a URL.

In order to measure the effect of caching, the workload was run under two different conditions for each of the web proxies. First, the cache was emptied before the run so that it did not contain any Internet documents. This case was referred to as the empty cache case. Second, the workload was run twice, once to store all of the documents in the cache and the second time to measure the average response time of the proxy. This case was referred to as the full cache case. These results are shown in figure 3.1.

From figure 3.1, it can be seen that the average response time of both of the tested web proxies was less for the full cache case then the empty cache case. This behavior was fully expected because in the full cache case, all of the requested documents were already stored in the cache and then could be returned to the user without having to download the entire contents of the document from the web server. The cache benefit which is the percent decrease of the average response time between the full and empty cache cases, shows that full cache can save about half of the average response time compared with empty cache.

| Avg. Response time | Empty cache | Full cache | Cache benefit |
|---|---|---|---|
| Traditional Proxy (Keep One) | 153.0 ms | 81.1 ms | 47% |
| Pachyderm (Keep All) | 197.8 ms | 85.7 ms | 57% |

Fig. 3.1: Average response time for keep one and keep all under empty cache case and the full cache case.

From Fig. 3.1, it can be noted that the average response time in both the full cache case and the empty cache case for Pachyderm is longer than the average response time for both of these cases performed by the traditional web proxy. It is hypothesized that this behavior was due to the overhead associated with the method used to organize Pachyderm's cache. In the more traditional web proxy, all the downloaded Internet documents are stored in one directory. However, in Pachyderm a different directory is created each time a new URL is requested by the user.

In order to test the hypothesis described above, code additions were placed into Pachyderm to measure the overhead associated with creating a directory every time the user requested a new Internet document. These results are shown in Fig. 3.2.

The difference in the average response time between Pachyderm and the traditional web proxy is 44.8 milliseconds in the case where the cache is empty. However, the difference in time to create the cache directories used in each of these web proxies was measured to be 45.1 milliseconds. From these measurements, it can be argued that the overhead observed in the first experiment was due to the creation of those directories used to store each new Internet document. To further support this claim, figure 3.2 shows that the average response time of the Pachyderm proxy verses the traditional proxy did not differ that much for the full cache case. This would be expected because Pachyderm does not create any directories in the full cache case.

| | Avg. Response Time | Directory Creation Time |
|---|---|---|
| Traditional Proxy (Keep One) | 153.0 ms | 25.8 ms |
| Pachyderm (Keep All) | 197.8 ms | 70.9 ms |
| Overhead | 44.8 ms | 45.1 ms |

Fig. 3.2 Pachyderm Overhead

### 3.2 Dependence of Average Response Time on Number of Documents in Cache

In order to see the combined effect of the benefit received from caching and the cost associated with creating a new directory for each downloaded Internet document, three different workloads were run on the traditional proxy and the Pachyderm proxy. Each of these runs consisted of two steps. First, a subset of an entire 10 URL workload was run which resulted in these documents being placed in the cache. Then, the entire 10 URL workload was run and the average response time to download the web documents was subsequently measured. Those results are shown in figure 3.3.

There are two important ideas that can be deduced from looking at figure 3.3. First of all, the overhead associated with Pachyderm decreases as the number of requested documents that are found in the cache increases. This behavior is expected since the number of new directories that must be created to store new Internet documents decreases. Secondly, when all of the documents are stored in cache, the performance of the traditional proxy and Pachyderm are very similar. When Pachyderm does not have to create any new directories, the performance very closely matches the performance of the traditional proxy.

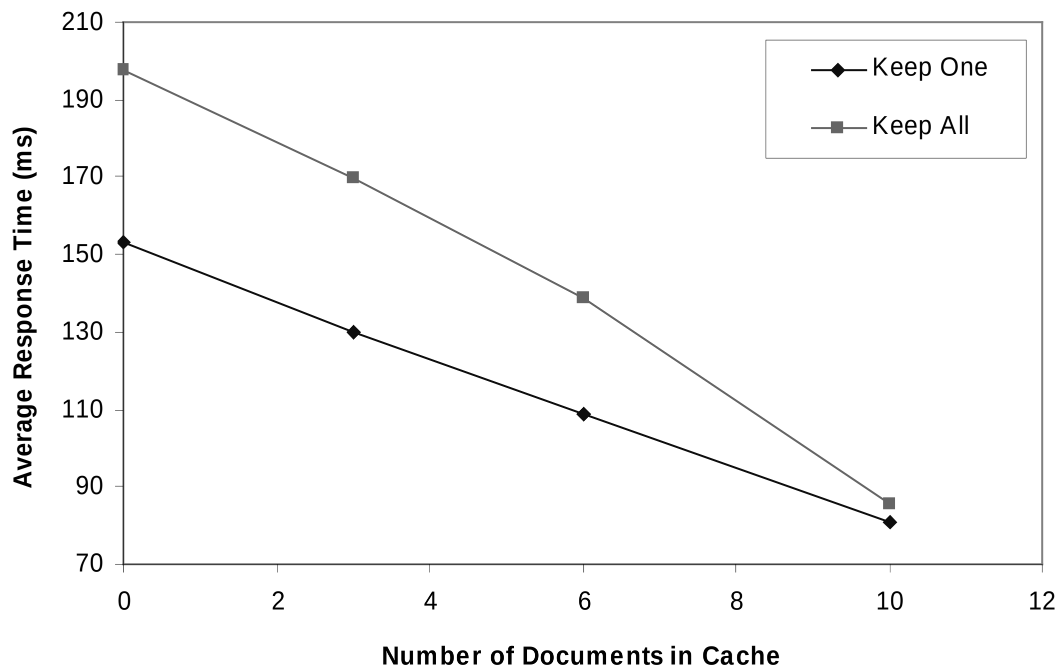


Fig. 3.3 Dependence of the average response time on number of documents stored in Cache.

### 3.3 GLIMPSE Indexing Overhead

In order to be able to search Pachyderm's cache of web pages with GLIMPSE, first an index has to be built on the cache. From figure 3.4, it can be seen that there is an overhead associated with indexing these documents. In order to index a 1.5 MB cache it took approximately 1.25 seconds. The amount of time to index the cache will increase as the cache size increases. However, for this application it is argued that a few seconds or even a minute will be accepted by the user in order to have the ability to find information that may no longer exist on the Internet. During normal Internet use, a user may have to wait several minutes due to network traffic or busy web servers for service; therefore, users have demonstrated the willingness to wait for information of interest from the Internet.

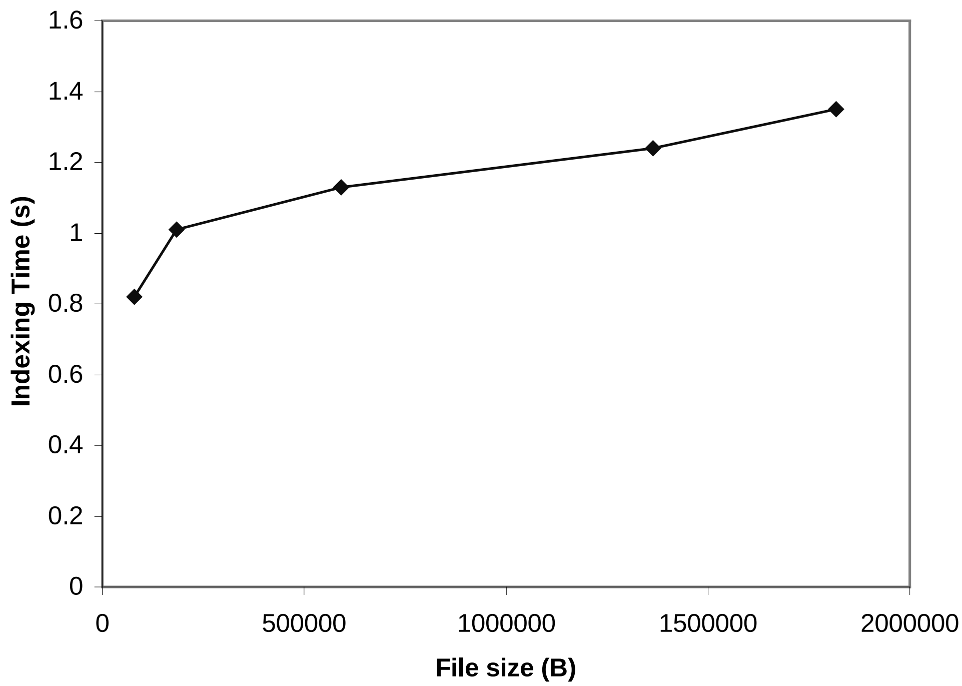


Fig. 3.4 GLIMPSE Indexing Time

## 4. Related Work

Web proxies and web page caching have been very active areas of recent research. The main goal of the various efforts in these areas has been to decrease the amount of network traffic and to increase the efficiency and reliability of the Internet.

Many studies have focused on hierarchical caching. In this caching method, proxy servers are connected in a hierarchical way. If the proxy server does not have a copy of the desired web page, it will query other web caches for that page. Squid [9] is a web caching proxy that uses the Internet Cache Protocol (ICP) to query external caches of Internet web pages. IPC provides a method of inter-cache communication that allows a cache to send messages to other caches asking for a particular web page. The external caches will examine their contents and will return the page if it is found. Cache hierarchies are a very flexible and scalable cache architecture; however, the messages necessary to query external caches consumes network bandwidth.

In order to reduce the number of messages sent, external cache summaries could be stored in the main memory. These main memory data structures would act as a hint system. The summary would indicate which external caches would most likely contain the desired web page. The proxy can then use this information to send targeted queries to the cache that will most likely have the desired page. This efficient cache querying technique was introduced in the Summary Cache system [3]. This method makes hierarchical caching more efficient and scalable than Squid does because the amount of network traffic it produces is smaller than the network traffic produced by Squid.

Currently, Pachyderm is only able to query its own local cache. Therefore, it has no need for a facility to communicate with other caches; however, in the future Pachyderm might be extended to deal with hierarchical caches.

## 5. Future Work

The Pachyderm web proxy has proven to be a method that not only can speed up web surfing, but also has the added ability to allow the user to view and search Internet pages that have been seen in the past. However, more functionality will needed to be added to the Pachyderm web proxy.

The current version of Pachyderm only supports the HTTP get and conditional get methods. In the HTTP version 1.1 protocol, there are many more methods that need to be supported including the put, post, head, delete and trace methods. Also, there are many other status codes returned by the response header that need to be specially dealt with, other than the Not-Modified status code that Pachyderm currently checks.

Along with making the Pachyderm web proxy fully HTTP version 1.1 protocol compliant, investigation into increasing the efficiency in which Internet documents are downloaded from the Internet could be done. Currently, each byte of information received from the web server is copied into an array of bytes one at a time, that entire array is then dumped to the client and the cache. It would be much more efficient if a new technique could be devised to copy groups of bytes from the web server into the byte array.

Finally, from the performance results found, there is a very high overhead associated with creating directories for each unique Internet web document downloaded by the user. This overhead can be decreased if the number of directories created within the cache is decreased. In order to achieve this goal, it would be necessary to rewrite the Pachyderm to only create one directory per day in which all of the Internet documents for that particular day would be stored. This method would decrease the number of directories created and is predicted to increase the performance so that it would be similar to the keep one cache that was implemented in this project for performance testing reasons.

## 6. Conclusions

Internet web proxies have been created to increase the efficiency and reliability of the Internet. It should also be noted that it is time that web proxies also give some stability to the Internet. Since the Internet is a dynamic environment, an Internet document that is viewed at one time might be changed or deleted the next time a user tries to find it. To solve this problem, the authors have created and introduced the Internet web proxy named Pachyderm. Pachyderm is a web proxy that not only stores the most recently viewed Internet documents, but it also stores all of the past versions of viewed cacheable Internet documents. This methodology has become more feasible in the last few years because storage is getting larger and cheaper; consequently, historical information can be stored at a low cost.

This paper has presented arguments for an Internet web proxy that stores historical information. It has been shown that this type of functionality can be provided without a high overhead cost and very little performance penalty over a more traditional web proxies.

## 7. References


1. C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, M. F. Schwartz, and D. P. Wessels. Harvest: A Scalable, Customizable Discovery and Access System. Technical report, Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado - Boulder, March 1995.

2. Edith Cohen, Balachander Krishnamurthy, and Park Avenue. Imporving End-to-End Performance of the Web Using Server Volumes and Proxy Filters.

3. Li Fan, Pei Cao, and Jussara Almeida. Summary Cache: A Scalable Wide-Area Cache Sharing Protocol. URL:http://www.cs.wisc.edu/~cao/papers/sumarycache.html.

4. C. Maltzahn, K. J. Richardson, and D. Grunwald. Performance Issues of Enterprise Level Web Proxies. In Proc. Of the 1997 SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Jun 1997. ACM.

5. Udi Madber, Sun Wu. GLIMPSE: A Tool to Search Through Entire File Systems. URL http://webglimpse.org/pubs/glimpse.ps.

6. Robert Olofsson, Fredrik Widlert. RabbIT2. URL http://www.nada.kth.se/projects/prup98/web_proxy/eng_index.html.

7. Marc J. Rochkind. The source code control system. IEEE Transactions on Software Engineering, 1(4):364-370, December 1975.

8. D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R. W. Carton, and J. Ofir. Deciding when to forget in the Elephant file system. Operating Systems Review, 34(5): 110-123, December 1999.

9. Duane Wessels. ICP and the Squid Web Cache. August 1997.